IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR UNITED STATES PATENT

FOR

# KNOWLEDGE MANAGEMENT SYSTEM FOR COMPUTER-AIDED DESIGN MODELING

Inventors:

David W. Vredenburgh

Gregory J. Smith

Robert J. Mattarn

Attorney Docket No.: 2834/101

Attorneys:

BROMBERG & SUNSTEIN LLP
125 Summer Street
Boston, MA 02110
(617) 443-9292

# KNOWLEDGE MANAGEMENT SYSTEM FOR COMPUTER-AIDED DESIGN MODELING

## FIELD OF THE INVENTION

The present invention relates generally to computer-aided design, and, more particularly, to a knowledge management system for computer-aided design modeling.

## BACKGROUND OF THE INVENTION

Computer-aided design (CAD) systems can be used to produce and manipulate geometric models. CAD systems often include very sophisticated algorithms for producing complex geometric models, manipulating those models, and analyzing the components of those models. For example, a CAD system may be used to model a complex structure having a curved surface, view the structure at various angles, and calculate the surface area of the curved surface and the volume of the structure as a whole. It might be useful to know the surface area of the curved surface, for example, to determine how much paint would be needed to cover the surface. It might be useful to know the volume of the structure, for example, to determine how much material would be needed to produce the structure.

While CAD systems can be very powerful modeling tools, they are generally limited to geometric modeling. Thus, CAD systems generally require the user to apply product design rules and practices necessary to produce the model. For example, if the user is required by an employer to use certain practices (such as, for example, always using a certain size bolt to connect two components), then the user must apply those practices to the model. The user is typically also required to make successive changes to a model when changing a component of the model. For example, each time the user changes an attribute of a component (such as, for example, the outside diameter of a component), the user may have to change attributes of one or more other components that connect or otherwise interact with that component, and the effects of these changes may cascade through many components of the model. The user is typically also required to handle non-geometric attributes of the model (such as, for example, component pricing

and manufacturing processes). As a result of these limitations, CAD systems can be difficult to use, particularly for casual CAD users (such as engineers, architects, or managerial staff) who may not be proficient with the CAD system but often need to make modifications to drawings or models on an as-needed basis.

Knowledge-based engineering (KBE) attempts to combine some level of knowledge management with design automation. Knowledge management typically includes such things as best practices, lessons learned (e.g., from earlier models), common practices (e.g., industry standards, company policies), product design rules, and quality metrics. Knowledge management might be applied to design automation, for example, to reduce the number of parts a company needs to order (e.g., by reusing parts from one model in another model), reduce design time, reduce product cost, and produce higher quality and reliability. KBE functionality is typically implemented within a CAD system or as an add-on to a CAD system (e.g., as a plug-in) so as to provide the CAD system with additional knowledge management capabilities.

CAD systems are often used in conjunction with computer-aided engineering (CAE) analysis tools for performing advanced model analysis. CAD systems are also often used in conjunction with product document management (PDM) tools for generating and maintaining product documentation. These CAE and PDM tools can be implemented as stand-alone applications or as add-ons to a CAD system. The KBE functionality may interact with the CAE and PDM tools to gather or provide information.

## SUMMARY OF THE INVENTION

In various embodiments of the present invention, a knowledge management system captures, stores, manages, and applies rules for modeling geometric objects and related non-geometric attributes. The knowledge management system controls a computer-aided design system for modeling a geometric structure, typically through an application program interface of the computer-aided design system. The knowledge management system typically includes a knowledge management application in communication with the computer-aided design system through an application program interface. The knowledge management system typically also includes a central database managed by a knowledge storage application for maintaining rules and other related

information. The knowledge management system typically also includes a knowledge acquisition application for generating rule programs for storage in the central database.

Rule programs are generated by the knowledge management system based on information that can be provided by a user, imported from the computer-aided design system, or extracted from a product document management application. The rules can relate to both geometric and non-geometric attributes of the model. Non-geometric attributes can be dependent on geometric attributes. Geometric structures can be modeled based on rules relating to non-geometric attributes.

In certain embodiments of the present invention there is provided a computer-aided modeling system includes a knowledge management system for managing a set of modeling rules and a computer-aided design system controlled by the knowledge management system. The knowledge management system generates instructions for modeling a geometric structure based on the set of modeling rules and communicates the instructions to the computer-aided design system for generating a model of the geometric structure.

The knowledge management system can include a knowledge management application in communication with the computer-aided design system through an application program interface of the computer-aided design system. The knowledge management system can also include a knowledge storage application in communication with the knowledge management application for storing the set of rules in a central database and communicating the set of rules to the knowledge management application. The knowledge management system can also include a knowledge acquisition application in communication with the knowledge storage application for generating the set of rules and communication the set of rules to the knowledge storage application for storage in the central database. The knowledge management system can produce a graphical display on a graphical user interface, including a first portion including information from the knowledge management system and a second portion including information from the computer-aided design system (such as a display window including a graphical representation of the geometric structure, with or without controls for manipulating the graphical representation).

In other embodiments of the present invention there is provided a method for computer-aided design modeling involves generating instructions for modeling a geometric structure based on a set of modeling rules and communicating the instructions to a computer-aided design system for generating a model of the geometric structure. In certain embodiments of the present invention, the set of rules are obtained from a central database over a communication network. The computer-aided design system typically includes an application program interface, and the instructions are typically communicated to the computer-aided design system through the application program interface.

The method can also involve producing a graphical display on a graphical user interface, including a first portion including information relating to the set of modeling rules and a second portion including information from the computer-aided design system. This may involve directing a window display generated by the computer-aided design system to be displayed on the graphical user interface. The display window can include a graphical representation of the geometric structure, and may also include controls for manipulating the graphical representation of the geometric structure.

In other embodiments of the present invention there is provided apparatus for computer-aided design modeling, the apparatus includes a design modeler for generating instructions for modeling a geometric structure based on a set of modeling rules and an interface from the design modeler to a computer-aided design system for communicating the instructions to the computer-aided design system for generating a model of the geometric structure. The apparatus typically also includes an interface from the design modeler to a central database over a communication network for obtaining the set of modeling rules from the central database. The computer-aided design system typically includes an application program interface, and the interface from the design modeler to a computer-aided design system typically complies with the application program interface.

In certain embodiments of the present invention, the apparatus includes a graphical user interface. The design modeler can produce a graphical display on the graphical user interface, including a first portion including information from the design modeler and a second portion including information from the computer-aided design system. This may involve the design modeler directing a window display generated by

the computer-aided design system to be displayed on the graphical user interface. The display window can include a graphical representation of the geometric structure, and may also include controls for manipulating the graphical representation of the geometric structure.

In still other embodiments of the present invention there is provided a computer program for computer-aided design modeling can be embodied in a computer readable medium. The computer program includes means for generating instructions for modeling a geometric structure based on a set of modeling rules and means for communicating the instructions to a computer-aided design system for generating a model of the geometric structure. In certain embodiments of the present invention, the set of rules are obtained from a central database over a communication network. The computer-aided design system typically includes an application program interface, and the instructions are typically communicated to the computer-aided design system through the application program interface.

The computer program typically includes means for producing a graphical display on a graphical user interface, including a first portion including information relating to the set of modeling rules and a second portion including information from the computer-aided design system. This may involve directing a window display generated by the computer-aided design system to be displayed on the graphical user interface. The display window can include a graphical representation of the geometric structure, and may also include controls for manipulating the graphical representation of the geometric structure.

In other embodiments of the present invention there is provided apparatus for computer-aided design modeling including means for generating instructions for modeling a geometric structure based on a set of modeling rules and means for communicating the instructions to a computer-aided design system for generating a model of the geometric structure.

In the various embodiments of the present invention, the set of rules can include rules relating to both geometric and non-geometric attributes. Rules relating to non-geometric attributes can include, for example, a rule for determine a cost of the geometric structure or a rule for defining a process. The set of rules can include rules relating to a

class having a plurality of geometric structures. The set of rules can include rules relating to geometric structures defined in the computer-aided design system.

Typical embodiments include a three-dimensional computer-aided design system controlled by a knowledge management system, such as the SOLIDWORKS(TM) three-dimensional computer-aided design system, and may additionally include a two-dimensional computer-aided design system, such as the VISIO(TM) two-dimensional computer-aided design system.

## BRIEF DESCRIPTION OF THE DRAWINGS

In the accompanying drawings:

FIG. 1 is a block diagram showing an exemplary modeling system in accordance with an embodiment of the present invention;

FIG. 2 is a block diagram showing the relevant components of the knowledge management system in accordance with an embodiment of the present invention;

FIG. 3A is a block diagram showing relevant components of the CAD system in accordance with an embodiment of the present invention;

FIG. 3B is a block diagram showing the relevant components of a CAD program in accordance with an embodiment of the present invention;

FIG. 4 is a block diagram showing an exemplary computer-aided modeling system in accordance with an embodiment of the present invention;

FIG. 5 shows an exemplary user interface screenshot for importing information from the CAD system relating to a mounting assembly, such as might be generated by the knowledge management application in accordance with an embodiment of the present invention;

FIG. 6 shows an exemplary user interface screenshot for defining a new geometric specification relating to the Mounting part family, such as might be generated by the knowledge management application in accordance with an embodiment of the present invention;

FIG. 7 shows an exemplary user interface screenshot displaying information upon entry of the name of a new CAD part file, such as might be generated by the knowledge management application in accordance with an embodiment of the present invention;

FIG. 8 shows an exemplary user interface screenshot for defining a geometry feature, such as might be generated by the knowledge management application in accordance with an embodiment of the present invention;

FIG. 9 shows an exemplary user interface screenshot showing geometry features (properties) defined for the CAD part file, such as might be generated by the knowledge management application in accordance with an embodiment of the present invention;

FIG. 10 shows an exemplary user interface screenshot for defining a mating, such as might be generated by the knowledge management application in accordance with an embodiment of the present invention;

FIG. 11 shows an exemplary user interface screenshot showing mating definitions, such as might be generated by the knowledge management application in accordance with an embodiment of the present invention;

FIG. 12 shows an exemplary user interface screenshot showing a rule for determining a fan area for the fan, such as might be generated by the knowledge management application in accordance with an embodiment of the present invention;

FIG. 13 shows an exemplary user interface screenshot including an embedded graphical representation of a model generated by the CAD system, such as might be generated by the knowledge management application in accordance with an embodiment of the present invention;

FIG. 14 shows a first exemplary user interface screenshot including a sub-window generated by the CAD system, such as might be generated by the knowledge management application in accordance with an embodiment of the present invention;

FIG. 15 shows a second exemplary user interface screenshot including a sub-window generated by the CAD system, such as might be generated by the knowledge management application in accordance with an embodiment of the present invention;

FIG. 16 shows a third exemplary user interface screenshot including a full view window generated by the CAD system, such as might be generated by the knowledge management application in accordance with an embodiment of the present invention;

FIG. 17 shows an exemplary user interface screenshot including a window generated by a two-dimensional CAD system, such as might be generated by the

knowledge management application in accordance with an embodiment of the present invention;

FIG. 18 shows an exemplary user interface screenshot including an analysis window, such as might be generated by the knowledge management application in accordance with an embodiment of the present invention;

FIG. 19 shows the relationship between the user (engineer), knowledge management application, knowledge database, and the integrated systems controlled by the knowledge management application in accordance with an embodiment of the present invention; and

FIG. 20 shows the relationship between the knowledge management application and the integrated systems in greater detail;

FIG. 21 is a logic flow diagram showing exemplary logic for class-based rules in accordance with an embodiment of the present invention; and

FIG. 22 is a logic flow diagram showing exemplary logic for the knowledge management application in accordance with an embodiment of the present invention.

## DETAILED DESCRIPTION OF A PREFERRED EMBODIMENT

For the purpose of the present description, the attachments thereto, and the accompanying claims, the following terms shall have the indicated meanings unless the context otherwise requires:

**ActiveX Control** - A reusable control written in Visual Basic used to create custom user interfaces in nAct.

**Application** – In the context of a model, a unit of deployment that contains one or more top-level parts and its components. In some cases, top-level part families can reference components in other applications.

**Assembly** - A grouping of parts and their subparts. An assembly is the hierarchy of parts that make up something meaningful to the engineers, such as a top-level part or a finished good.

**Calculated Value** - The value that was calculated by the formula that was entered into the rules database. The formula may return a constant, a calculation based on other properties or a value obtained from a customer database.

**Check In / Check Out** –

- A project cannot be deleted if it contains line items that are checked out to another user.
- A line item cannot be deleted or edited if it is checked out to another user.
- A model cannot be edited if the line item is checked out to another user.
- When a user edits a model that is not checked out to anyone, the corresponding line item automatically becomes checked out to them.
- Line items can be manually checked in or out by clicking the Check In and Check Out buttons beneath the line item grid or by clicking the corresponding options under the Line Item menu. While editing a model automatically checks out a line item, the check in process must always be done manually.
- When a line item is checked out, Checked_Out_By column in the LineItem table in the Project database is updated with the user's username. When the line item is checked back in, the column is set to null.
- Anyone can edit a line item (or its model) if the line item is not checked out to anyone.

**Child Part Family** - A family that inherits properties, subparts and connections from another part family (template part family).

**Code Generation** - The creation of rules into executable source code for use in nAct during run-time. The generated code can also be compiled into a dynamic link library (.DLL) using Visual Basic.

**Column** - In an SQL table, the area in each row that stores the data value for some attribute of the object modeled by the table.

**Compiled Mode** – Editing models in nAct with rules that are compiled into a DLL file using Microsoft Visual Basic 6.0. In a typical embodiment of the invention, compiled rules cannot be seen by the user and cannot be debugged at runtime. (See also Developer Mode)

**Component** - A component is any part or sub-assembly within an assembly.

**Connection Collection** - A conceptual or physical relationship between two or more parts. This differs from a subpart collection in that the two connected parts do not have a parent-child relationship (such as a nut and a bolt).

**Connection Constraint** - The formula used to determine the correct part or parts to be connected at run-time. There is preferably never a connection constraint associated with a connection specification where the same part family owns both.

**Connection Point** – A point in which two schematic shapes can be connected directly or via wire.

**Connection Spec** - The definition of a relationship between two parts at run-time (e.g. a hose and a valve). A connection exists when one part in an assembly needs to reference the properties of another part in an assembly and the user cannot or prefers not to walk up and down the part/subpart hierarchy.

**Constraint** - A formula that indicates how a specification should be computed in the design model.

**Context** - The current location of the part in the tree defined by its parents, grandparents, etc. A part may be used in more than one location in the tree, and each location is considered a different context.

**Custom Control** – A customer-developed ActiveX Control written in Microsoft Visual Basic 6.0 and compiled to an .OCX file. Custom controls are used to create custom interfaces for process steps.

**Database** – A collection of data arranged for ease and speed of search and retrieval. A database also includes an engine which accepts commands from the user to insert, update, delete and select the data. Exemplary embodiments of the invention use Microsoft SQL Server 2000 database to store its data.

**Decache** – Resetting values based on a dependency.

**Dependency** – A property or attribute that is used in the calculation of another formula.

**Design Time** – Using nAct Expert to create part families and specifications (properties, subparts, connections, database constraints, geometry and schematics) and to write formulas for rules. (See also Run Time)

**Developer Mode** – Editing models in nAct with rules running in source code using Microsoft Visual Basic for Applications. The user can see the actual formulas for the rules and can step through and debug them at run time. (See also Compiled Mode)

**Display Name** - This is a property that is used to create a formula-driven name that appears in the run-time application instead of the part family pretty name. (Display Name can be placed on any part family.)

**Drawing** - A Drawing is a 2D representation of a 3D part or assembly.

**Feature** - A geometric attribute of a part (i.e. plane, extrusion, sketch, cut, etc.) whose dimensions can be driven by nAct properties within the model.

**Follow Part Number** - When this flag is set, all parts with the same part number will share the user input value of its properties, subparts or connections.

**Frame** – A component of a process step layout. Each layout contains 1 to 4 frames. Each frame contains 1 of the following user interface elements: dynamically generated form, CAD system, 2D modeler, drawing, report or custom control.

**Hierarchy** - A logical tree structure that organizes the members of a dimension such that each member has one parent member and zero or more child members.

**IIS** – Microsoft Internet Information Services. IIS includes a web server, SMTP (mail) services and FTP (file transfer protocol) services. The RuleStream Enterprise Web tools must be installed on an IIS server. IIS is a component of Microsoft Windows 2000 Server and Microsoft Windows 2003 Server and can be installed from the Windows CD.

**Index** - In a relational database, a database object that provides fast access to data in the rows of a table, based on key values. Indexes can also enforce uniqueness on the rows in a table. SQL Server supports clustered and nonclustered indexes. The primary key of a table is automatically indexed. In full-text search, a full-text index stores information about significant words and their location within a given column.

**Input Value** - A value that is entered by the user in nAct during run-time. This takes precedence over any calculated values.

**Kernel** - The knowledge engine that executes rules. It provides the base system functionality for managing the run-time model.

**Layout** – The configuration of the user interface of the model designer window in nAct. Each step in the selected process can have its own unique layout. A layout

consists of 1 to 4 frames, and each frame can contain 1 of the following user interface elements: dynamically generated form, CAD system, 2D modeler, drawing, report or custom control.

**Line Item** - A top-level part that is associated with a project. The line item has an associated top-level part family.

**Local Constraint** - A constraint owned the same part family as the associated spec. This is the default formula.

**Model** - An actual design created using nAct (the graphical representation of a line item).

**nAct** - The client server tool used to create projects, line items and models.

**nAnswer** - The web-based tool which allows users to query the databases using English like query statements.

**nExplore** - The web-based tool used to view rules and documentation.

**nPlatform** - The project and rules repository.

**nQuality** - The web-based tool for entering and reporting quality data, scoring and risks.

**Override** - The action of a higher-level part within an assembly determining the value for a subordinate part.

**Owner Part Family** - The part family that contains the specification or constraint.

**Parent Part Family** - The part family that owns the subpart collection of which the part is a member.

**Part** - An instance of a part family.

**Part Family** - The definition of a fundamental object in an assembly. This may be a component or a finished good.

**Process** – A defined set of steps. Each top-level part family can have 1 or more processes. The processes that will be used during run time are determined by the line item definition.

**Process Step** – A defined workflow for editing a model during run time. Each process step has a layout that defines its user interface. Process steps are grouped in

categories and have a defined order within the process. Each process must have 1 or more process steps.

**Property Constraint** - The formula used to determine the value of a property.

**Property Specification** - The definition of the characteristics of a part family (e.g. height, weight, length).

**Release** – A snapshot of a model and its values.

**Row** - In an SQL table, the collection of elements that form a horizontal line in the table. Each row in the table represents a single occurrence of the object modeled by the table and stores the values for all the attributes of that object.

**Run Time** – Using the nAct Model Designer to edit a model using defined part families, specs and rules. Users design models in run time in Compiled Mode or Developer Mode. (See also Design Time)

**Schematic** – A 2D representation, usually associated with electrical components.

**Score** – A value assigned to rate the effectiveness of the design.

**Shape** – A 2D object within a schematic diagram.

**Specification** - The constituent elements of a part family. Specifications indicate that parts of that type always have certain properties, subpart collections, or connection collections.

**Step** – (see Process Step)

**Stored Procedure** - A precompiled collection of Transact-SQL statements stored under a name and processed as a unit.

**Subpart Collection** - A collection of component parts.

**Subpart Link Constraint** - The formulas used to determine existence, type, quantity and/ or names of parts within the collection.

**Subpart Link Spec** - Provides the valid part families that can become components at run-time.

**Subscription** - The ability of one part family to assume the property, subpart or connection definitions and formulas of another part family.

**Table** - A two-dimensional object, consisting of rows and columns, used to store data in a relational database. Each table stores information about one of the types of objects modeled by the database.

**Template Part Family** - The part family from which a child part family inherits its characteristics.

**UDF** (User Defined Fields) - Customizable columns on the project table. There are four UDF columns available.

**Unit** – A unit of measure for a numeric property value. Like units are grouped in unit categories. The unit category "Length" may include units such as "feet", "meters", "inches" or "millimeters".

**VB** – Microsoft Visual Basic 6.0. In exemplary embodiments of the invention, Visual Basic is the language used to create custom controls for use in process steps and to create compiled versions of rules (application DLLs).

**VBA** – Microsoft Visual Basic for Applications.

**View** - A database object that can be referenced the same way as a table in SQL statements. Views are defined using a SELECT statement and are analogous to an object that contains the result set of this statement.

**Wire** - A line or connector used when two connected objects are not coincidental.

In embodiments of the present invention, a knowledge management system captures, stores, manages, and applies rules for modeling geometric objects and related non-geometric attributes. The knowledge management system preferably supports rules relating to geometric attributes that can be represented and manipulated by a CAD system as well as rules relating to various other attributes that generally cannot be represented and manipulated by the CAD system, such as certain "negative" geometric attributes (e.g., a specification for producing a hole in a component or for removing material from a component so as to form a feature of that component), certain geometry-based attributes that are not modeled as physical features of a structure or assembly, and non-geometric attributes (e.g., pricing, processes, component relationships, component classes, user intentions, and abstractions). The rules are typically stored in a central database so that the rules can be manipulated independently of any modeling. The rules can be derived from various sources, including general engineering principles, user-provided information, and information obtained from a PDM system. The knowledge management system can tracks rule changes over time, and can apply a particular version of the rules

to a model. The knowledge management system defines the components and parameters for a particular model based on the rules being applied. The knowledge management system can incorporate predefined components (such as components created in a CAD system) or dynamically defined components into the model.

The knowledge management system is generally independent of the CAD system, although the knowledge management system can interface with a CAD system for, among other things, importing/integrating component specifications from the CAD system for use in modeling, instructing the CAD system to produce a geometric model incorporating the components and parameters defined by the knowledge management system, and obtaining geometric information relating to a model (e.g., sizes, surface areas, volumes) for use by the knowledge management system (e.g., for computing the cost of a component based on its volume according to pricing rules provided by a user). A geometric model produced by the CAD system can be viewed and manipulated as usual using the CAD system, although it is preferable for any and all changes to the model to be coordinated through the knowledge management system so that the appropriate rules can be applied to the changes.

In certain embodiments of the present invention, the knowledge management system interfaces with the CAD system through an application program interface (API) of the CAD system. Specifically, the CAD system includes an API through which certain functions of the CAD system can be performed. The CAD system API is typically used for such things as macros (i.e., programs that perform a series of function steps for the user) and add-ons (i.e., programs that add functionality to the CAD system). In embodiments of the present invention, however, the knowledge management system uses the API to control the CAD system such that the CAD system runs only when activated by the knowledge management system. The knowledge management system typically activates the CAD system for such things as displaying a model to the user and obtaining geometric information relating to model components.

FIG. 1 is a block diagram showing an exemplary modeling system 100 in accordance with an embodiment of the present invention. Among other things, the modeling system 100 includes a knowledge management system 110 in communication with a CAD system 120. The knowledge management system 110 controls the CAD

system 120, specifically by generating instructions for modeling a geometric structure based on a set of modeling rules and communicating the instructions to the computer-aided design system 120 for generating a model of the geometric structure. The knowledge management system 110 can also interface with a CAE application 130 for analysis and with a PDM application 140 for product document management.

FIG. 2 is a block diagram showing the relevant components of the knowledge management system 110 in accordance with an embodiment of the present invention. Among other things, the knowledge management system 110 includes a knowledge acquisition application 210 for capturing and generating rules, a knowledge storage application 220 for storing rules and models in a central database 240, and a knowledge management application 230 for generating models based on the rules. Among other things, the knowledge acquisition application 210 generates rule programs based on information obtained from a user and communicates the rule programs to the knowledge storage application 220 for storage in the central database 240. The knowledge management application 230 obtains rule programs from the knowledge storage application 220 and applies the rule programs for building a model. The knowledge management application 230 may communicate model information to the knowledge storage application 220 for storage in the central database 240.

More specifically, the knowledge acquisition application 210 captures modeling rules and generates rule programs for storage by the knowledge storage application 220. The knowledge acquisition application 210 interacts with a user through a user interface through which the user enters information regarding components, design processes, engineering and manufacturing rules, and customer and marketing requirements. The knowledge acquisition application 210 generates rule programs from the user information, and sends the rule programs to the knowledge storage application 220 for storage. The knowledge acquisition application 210 allows rules to be modified quickly and easily.

The knowledge storage application 220 stores modeling information in a relational database, including, among other things, rule programs generated by the knowledge acquisition application 210 and models generated by the knowledge management application 230. The knowledge storage application 220 also tracks

revision histories, design status, and user group security. Multiple revisions of a rule can be stored so that a particular version of a rule can be applied to a model while another version of the rule is being created or modified. A design can be modeled using an earlier version of a rule if desired.

The knowledge management application 230 applies rules to create detailed computer models. The knowledge management application 230 can model two-dimensional schematics or three-dimensional geometries. The knowledge management application 230 can also model a bill of materials for the components of an assembly. The knowledge management application 230 can automatically update a previously completed model under revised or new rules. In this way, models can be generated by manipulating the rules through the knowledge management system rather than manipulating the model itself through the computer-aided design system.

FIG. 3A is a block diagram showing relevant components of the CAD system 120 in accordance with an embodiment of the present invention. Among other things, the CAD system 120 includes a 3D CAD program 301, such as such as SOLIDWORKS(TM) from SolidWorks Corporation, 300 Baker Avenue, Concord, MA 01742, and may also include a 2D CAD program 302, such as VISIO(TM) from Microsoft Corporation. Each of the CAD programs has its own API through which it interacts with the knowledge management application. FIG. 3B is a block diagram showing the relevant components of a CAD program, such as the 3D CAD program 301 or the 2D CAD program 302, in accordance with an embodiment of the present invention. Among other things, the CAD program includes a CAD application 320 having an API 310 through which certain functions of the CAD application 320 can be controlled. The interactions between the knowledge management application and different CAD systems depend to a large degree on the CAD system API. The SOLIDWORKS(TM) three-dimensional CAD program has an internal macro/recording language and also supports an application programming interface language that is accessible through an OLE (Object Linking and Embedding) interface with support for VISUAL BASIC(TM) and VISUAL C++(TM). In an exemplary embodiment of the present invention, the knowledge management application is substantially reactive to the VISIO(TM) two-dimensional CAD system, while the knowledge management application actively controls the SOLIDWORKS(TM) three-

dimensional CAD system. Thus, support for each particular CAD system generally requires some level of integration by the knowledge management application to work with the specific functions and API of the CAD program.

In certain embodiments of the present invention, the various functions of the knowledge management system 110 are divided among different devices that communicate over a communication network, such as the public Internet or public or private intranets. In an exemplary embodiment of the present invention, knowledge storage functions reside in one or more storage servers that incorporate the knowledge storage application 220 and central database 240, while knowledge acquisition and management functions reside in user workstations (such as personal computers) or terminals that incorporate the knowledge acquisition application 210, the knowledge management application 230, and the CAD system 120. The user workstations or terminals typically include a user interface for interacting with a user and a network interface for communicating with the storage server over a communication network.

FIG. 4 is a block diagram showing an exemplary computer-aided modeling system in accordance with an embodiment of the present invention. Among other things, the system 400 includes one or more user workstations 410 in communication with one or more storage servers 430 over a communication network 420. The workstation 410 incorporates the knowledge acquisition application 210, the knowledge management application 230, and the CAD system 120, and also includes a user interface for interacting with the user and a network interface for communicating over the communication network 420 with the storage server(s) 430. The user interface 411 is typically a graphical user interface that provides for both displaying information to the user and receiving inputs from the user. The storage server 430 incorporates the knowledge storage application 220 and the central database 240, and also includes a network interface 431 for communicating over the communication network 420 with the user workstation(s) 410.

Within the user workstation 410, the knowledge acquisition application 210 and the knowledge management application 230 interact with the user through the user interface 411, and also interact with the knowledge storage application 220 in the storage server 430 through the network interface 412 using a client-server paradigm. The

knowledge management application 230 also controls the CAD system 120 through an API of the CAD system 120. The user workstation 410 is typically a general-purpose computer, and the knowledge acquisition application 210, the knowledge management application 230, and the CAD system 120 are typically software programs that run on the general-purpose computer.

Within the storage server 430, the knowledge storage application 220 interacts with the central database 240 through a database interface (not shown), and interacts with the knowledge acquisition application 210 and the knowledge management application 230 in the user workstation 410 through the network interface 431. The storage server 430 is typically a general-purpose computer, and the knowledge storage application 220 is typically a software program that runs on the general-purpose computer. The central database 240 is typically a relational database.

FIG. 19 shows the relationship between the user (engineer) 1910, knowledge management application 1920, knowledge database 1930, and the integrated systems 1940 controlled by the knowledge management application 1920, including CAD system(s) and possibly also a CAE application, a PDM application, and a component databases. The knowledge management application 1920 extracts rules for design automation from the knowledge database 1930. The knowledge management application 1920 may also receive specifications, rules, and other information relating to modeling. A product control modeler element of the knowledge management application 1920 interacts with the integrated applications as necessary for modeling, analysis, product document management, and component selection. Information generated by the knowledge management application 1920, such as runtime rule authoring and trend analysis, may be stored in the knowledge database 1930.

FIG. 20 shows the relationship between the knowledge management application and the integrated systems in greater detail. The knowledge management application 1920 interacts with the CAD system 2010 for modeling such things as features, mating conditions, surface area, volume, and mass properties. The knowledge management application 1920 interacts with the CAE analysis application 2020 for such things as stress, thermal, kinematics, and loads analysis. The knowledge management application 1920 interacts with the PDM application 2030 to generate and utilize such things as files,

structure, workflow, and bill of materials (BOM). The knowledge management application 1920 interacts with component databases 2040 for such things as part numbers, standards, inventory, and pricing.

In typical embodiments of the present invention, a model may include multiple geometric components. Each component can be associated with both geometric attributes and non-geometric attributes. Rules can be established for defining relationships between components without necessarily defining that actual parameters of the relationship (such as, for example, a rule that a fan blade assembly must mate with a motor shaft, without necessarily defining the shape or size of the shaft which might affect the type of mating). Components can be organized into classes (such as, for example, three possible motors for a fan assembly can be organized into a "motors" class), and rules can be established for the class as a whole such that the rules are applied to whatever class member is selected for inclusion in a particular model (such as, for example, a generic rule that any of the class of motors must mate with a fan blade component). Rules can be established for selecting a particular member of a class for a particular model (such as, for example, a rule for selecting a particular motor based on the amount of power or the rotational speed required for a model, or a rule for selecting the number of fan blades for the fan blade component based on the volume of air to be moved and other parameters such as the motor selected and the diameter of the fan blade component). Rules relating to "negative" attributes can be defined (such as, for example, a rule that a motor frame must include a hole in a particular location for bolting the motor frame to a chassis) and applied to a model component as a library feature. Rules relating to various non-geometric attributes can be established (such as, for example, rules for deriving manufacturing processes based on the components incorporated into a selected model, or rules for estimating component, sub-assembly, product, and manufacturing costs).

One particular implementation of a knowledge management application in accordance with an embodiment of the present invention is described in Attachment I (consisting of numbered pages 1-91) and in Attachment II (consisting of numbered pages 1-113), each of which is hereby incorporated herein by reference in its entirety. Within these attachments, the knowledge management system may be referred to generally as "Rulestream" or "Navion," the knowledge management application may be referred to as

"nAct" or "nAct Engineer," the knowledge acquisition application may be referred to as "nAct Expert," and the knowledge storage application may be referred to as "nPlatform." In Attachment II, edit rules relating to "Geometry Specification" and "Geometry Constraint" generally relate to the SOLIDWORKS(TM) three-dimensional CAD system, while edit rules relating to "2DSchematic Specification" generally relate to the VISIO(TM) two-dimensional CAD system.

As discussed above, the knowledge management application controls the CAD system through a CAD system API. While the actual CAD functions that can be performed through the API are substantially limited by the API (and are subject to change by the CAD system provider), the specific API functions used and the manner in which the API functions are used are determined by the knowledge management application for performing specific knowledge management operations. In an exemplary embodiment of the present invention, the knowledge management application controls the SOLIDWORKS(TM) three-dimensional CAD system through its API. In order to control the CAD system, the knowledge management application typically performs such operations as starting SOLIDWORKS(TM), opening a part file, opening an assembly file, mating a component, deleting a mate, fixing a component, removing a part or assembly, suppressing a component, hiding/showing a component, suppressing a feature, inserting a library feature, removing a library feature, setting a part dimension, setting an assembly dimension, creating a component pattern, removing a component pattern, creating a feature pattern, removing a point in a sketch, removing a feature pattern, setting a custom property, setting component color, and closing SOLIDWORKS(TM). This is not meant as an exhaustive list, and the knowledge management application can perform other operations as needed. Exemplary API calls and settings for performing the above operations in an exemplary embodiment of the invention are described below.

Starting SOLIDWORKS(TM) may involve use of the following API functions:

- Set SW = New SldWorks.SldWorks
- SW.UserControl = False
- Set Assembly = SW.OpenDoc6(strFilename, swDocPART, swOpenDocOptions_Silent, "", lngErr, lngMess)

The following SOLIDWORKS(TM) settings may be used:

swMateAnimationSpeed = 0
swLargeAsmModeAutoActivate = swResponseNever
swPerformanceAssemRebuildOnLoad = swResponseAlways
swLoadExternalReferences = swResponseNever
swAutoSaveInterval = 0
swBackupCopiesPerDocument = 0
swShowErrorsEveryRebuild = False
swMaximizeDocumentOnOpen = True
swSnapToPoints = False
swLargeAsmModeAutoLoadLightweight = False
swLargeAsmModeUpdateMassPropsOnSave = False
swLargeAsmModeAutoRecover = False
swAutoLoadPartsLightweight = False
swPerformanceVerifyOnRebuild = False
swEnablePerformanceEmail = False
swUseFolderSearchRules = False
swExtRefUpdateCompNames = True
swFeatureManagerEnsureVisible = False

Opening a part file typically involves opening the part file, adding the part file as a component to a parent assembly, closing the part file, and rebuilding the top level assembly. The following API functions may be used:

- Set objDoc = SW.OpenDoc6(strFilename, swDocPART, swOpenDocOptions_Silent, "", lngErr, lngMess)
- Assembly.AddComponent2(strFilename, 0, 0, 0)
- SW.CloseDoc objDoc.GetTitle
- Assembly.EditRebuild3

Opening an assembly file typically involves opening the part file, adding the part file as a component to a parent assembly, closing the part file, and rebuilding the top level assembly. If assembly dimensions are driven by the knowledge management application, then all components are typically renamed to ensure uniqueness. The following API functions may be used:

- Set objDoc = SW.OpenDoc6(strFilename, swDocPART, swOpenDocOptions_Silent, "", lngErr, lngMess)
- Set objConfiguration = objDoc.GetActiveConfiguration()
- Set objComponent = objConfiguration.GetRootComponent()
- objComponent.GetChildren
- Set objChildDoc = objChild.GetModelDoc
- objChildDoc.SaveAs4 strNewFileName swSaveAsCurrentVersion, swSaveAsOptions_Silent, lngErr, lngWarnings
- SWAssembly.AddComponent2(strFilename, 0, 0, 0)
- SW.CloseDoc objDoc.GetTitle
- SWAssembly.EditRebuild3

Mating a component typically involves putting the parent assembly in "edit" mode, selecting the features to mate, adding the mate, and rebuilding the assembly. The mate name is then found and noted by the knowledge management application in the case where a dependent property is changed and the mating is effected. The following API functions may be used:

- ObjParentComponent.Select False
- Assembly.EditAssembly

*Selecting Plane, Axis, or Point*

- Assembly.SelectByID strFeatureName, strFeatureType, 0, 0, 0

*(also used, strFeatureName & "@" & strComponentPath, and "Point1@" & strFeatureName)*

*Selecting Face or Edge (loop through Faces, or Faces and Edges to find name match)*

- objComponent.GetBody(), objBody.GetFirstFace(), objBody.GetNextFace(), objFace.GetEdges, Assembly.GetEntityName(obj)

- Assembly.AddMate lngMateType, lngAlignType, boolFlip, dblDist, dblAngle
- Assembly.EditRebuild3

*Finding the Mate created (find the MateGroup, move to the last SubFeature)*

- Assembly.FeatureByPositionReverse(i)
- objFeature.GetTypeName = "MateGroup"
- objMateGroup.GetFirstSubFeature
- objMate.GetNextSubFeature()

Deleting a mate typically involves selecting the mate using the parent assembly's model document and deleting the selection. The following API functions may be used:

- Assembly.SelectByID strMateName, "MATE", 0, 0, 0
- Assembly.DeleteSelection False
- Assembly.EditRebuild3

Fixing a component typically involves setting the component transform, selecting the component, and fixing the component. The following API functions may be used:

- objComponent.GetXform
- objComponent.SetXform (varXForm)

- objComponent.Select False
- Assembly.FixComponent
- Assembly.EditRebuild3

Removing a part or assembly typically involves selecting the parent assembly, putting the parent assembly in "edit" mode, selecting the component, and deleting the selection. The following API functions may be used:

- objParentComp.Select False
- Assembly.EditAssembly
- objComponent.Select False
- Assembly.DeleteSelection False
- Assembly.ClearSelection
- Assembly.EditAssembly
- Assembly.EditRebuild3

Suppressing a component typically involves checking the suppression state of the component and setting the suppression state, if suppressing the document is saved. The following API functions may be used:

- objComponent.IsSuppressed
- Set oModelDoc = objComponent.GetModelDoc
- oModelDoc.Save3 swSaveAsOptions_Silent, lngErr, lngWarnings
- objComponent.Select False
- Assembly.EditSuppress2 **or** Assembly.EditUnSuppress2
- Assembly.EditRebuild3

Hiding or showing a component typically involves setting the hidden state appropriately. The following API functions may be used:

- objComponent.IsHidden(False)

- objComponent.Select False
- Assembly.ShowComponent2, Assembly.HideComponent2
- Assembly.EditRebuild3

Suppressing a feature typically involves traversing the model documents to find the named feature and setting its suppression state accordingly. The following API functions may be used:

- objModelDoc.FirstFeature
- objFeature.Name()
- objFeature.GetNextFeature()
- objFeature.IsSuppressed
- objFeature.SetSuppression 0
- objFeature.SetSuppression 2

Inserting a library feature typically involves selecting the component to receive the feature, putting the component in "edit" mode, selecting the references required for insertion, and inserting the library feature. The new feature and its sub-features are typically renamed. The following API functions may be used:

- objParentComponent.Select2 False, 0
- Assembly.EditPart2 True, True, lngErr
- obj.Select2 True, intMark *(Traverse features and select for Edges and Faces)*
- Assembly.AndSelectByMark(strFeatureName & "@" & strComponentPath, strFeatureType, 0, 0, 0, intMark) *(Plane, Point, or Axis)*
- Assembly.InsertLibraryFeature(strFileName)
- Assembly.SelectedFeatureProperties 0, 0, 0, 0, 0, 0, 0, 1, 0, strNewName)
- objFeature.GetFirstSubFeature, subFeature.Name(), subFeature.GetNextFeature()

- Assembly.EditAssembly
- Assembly.EditRebuild3

Removing a library feature typically involves selecting component owning the feature, putting the component in "edit" mode, selecting the feature, and deleting the feature using the context of the top level assembly. The following API functions may be used:

- objComponent.Select2 False, 0
- Assembly.EditPart2 True, True, lngErr
- Assembly.ClearSelection
- Assembly.SelectByID strFeatureName & "@" & objComponentPath, "BODYFEATURE", 0, 0, 0
- Assembly.DeleteSelection False
- Assembly.EditAssembly
- Assembly.EditRebuild3

Setting a part dimension typically involves setting the read-only status of the dimension to false, setting the system value for the dimension, and resetting the read-only status of the dimension to true (this is because all dimensions controlled by the knowledge management application are preferably maintained as read-only to prevent modification through the CAD system). A dimension is typically references by a string (e.g. D1@Sketch1). The following API functions may be used:

- objDoc.Parameter(strDimension).ReadOnly = False
- objDoc.Parameter(strDimension).SystemValue = varValue
- objDoc.Parameter(strDimension).ReadOnly = True

Setting an assembly dimension typically involves all of the steps for setting an assembly dimension, except Parameter is additionally checked for existence on the components. The following API functions may be used:

- objDoc.Parameter(strDimension & "@" & strComponent).ReadOnly = False
- objDoc.Parameter(strDimension & "@" & strComponent).SystemValue = varValue
- objDoc.Parameter(strDimension & "@" & strComponent).ReadOnly = True

**-OR-**

- objDoc.Parameter(strDimension & "@" & strComponent & ".Part").ReadOnly = False
- objDoc.Parameter(strDimension & "@" & strComponent & ".Part").SystemValue = varValue
- objDoc.Parameter(strDimension & "@" & strComponent & ".Part").ReadOnly = True

**-OR-**

- objDoc.Parameter(strDimension & "@" & strComponent & ".Assembly").ReadOnly = False
- objDoc.Parameter(strDimension & "@" & strComponent & ".Assembly").SystemValue = varValue
- objDoc.Parameter(strDimension & "@" & strComponent & ".Assembly").ReadOnly = True

Creating a component pattern typically involves selecting the parent assembly, putting the parent assembly in "edit" mode, selecting the component and the feature pattern, and inserting the feature pattern. The Derived Pattern is typically renamed by traversing the parent assembly and finding the Derived Pattern using the seed component. The following API functions may be used:

- objComponent.Select False
- Assembly.EditAssembly
- Assembly.ClearSelection

- Assembly.SelectByID strComponentPath, "COMPONENT", 0, 0, 0
- Assembly.AndSelectByID strPatternName & "@" & strComponentPath2, "BODYFEATURE", 0, 0, 0
- Assembly.InsertDerivedPattern
- Set objFeature = objDoc.FirstFeature
- objFeature.GetTypeName = "DerivedSketchPattern"
- arrSeed = def.SeedComponentArray()
- arrSeed(i).Name
- def.ReleaseSelectionAccess
- Set objFeature = objFeature.GetNextFeature

Removing a component pattern typically involves selecting the Derived Pattern from the parent model document and deleting the selection. In some models, the referenced configuration may need to be reset to prevent future selections from failing on this part. The following API functions may be used:

- oParentDoc.ClearSelection
- oParentDoc.SelectByID strPatternName, "COMPPATTERN", 0, 0, 0
- oParentDoc.DeleteSelection False
- oParentComponent.ReferencedConfiguration = ""
- Assembly.EditRebuild3

The knowledge management application typically requires a part for each feature in the pattern, and maintains a property containing the X and Y coordinates for each feature. In order to create a feature pattern, the is activated and is used to select the Sketch Pattern. A point for each component required is created, it's identifier is stored, the Sketch is inserted, and the document rebuilt. The Sketch and Feature are selected and a Pattern is created. The Sketch Pattern is then found and renamed by traversing the Document and finding the Pattern using the seed feature. The following API functions may be used:

- SW.ActivateDoc2 objDoc.GetTitle, True, lngErr
- objDoc.ClearSelection
- objDoc.SelectByID strSketchName, "SKETCH", 0, 0, 0
- objDoc.SetAddToDB True
- objDoc.EditSketch
- Set oPoint = objDoc.CreatePoint2(lngX, lngY, 0)
- arrPointID = oPoint.GetId()
- objDoc.InsertSketch
- objDoc.EditRebuild3
- objDoc.SelectByID strSketch, "SKETCH", 0, 0, 0
- objDoc.AndSelectByID strFeature, "BODYFEATURE", 0, 0, 0
- objDoc.ActivateSelectedFeature
- objDoc.FeatureSketchDrivenPattern 1
- Set objFeature = objDoc.FirstFeature
- objFeature.GetTypeName = "SketchPattern"
- arrSeed = def.PatternFeatureArray()
- Set subFeature = arrSeed(i)
- objModelDoc.GetEntityName(subFeature) = strSeedName
- def.ReleaseSelectionAccess
- Set objFeature = objFeature.GetNextFeature
- SW.ActivateDoc2 Assembly.GetTitle, True, lngErr
- SW.CloseDoc objDoc.GetTitle

Removing a point in a sketch typically involves opening the document containing the sketch, putting the sketch in "edit" mode, traversing the sketch points to find the point with the corresponding identifier, selecting the point, and deleting the selected point. The following API functions may be used:

- SW.ActivateDoc2 objDoc.GetTitle, True, lngErr
- objDoc.ClearSelection
- objDoc.SelectByID strSketchName, "SKETCH", 0, 0, 0

- objDoc.EditSketch
- Set oSketch = objModelDoc.GetActiveSketch2()
- arrPoints = oSketch .GetSketchPoints
- arrID = arrPoints(i).GetId
- arrPoints(i).Select2 False, 0
- objDoc.DeleteSelection False
- objDoc.InsertSketch
- objDoc.EditRebuild3
- SW.ActivateDoc2 Assembly.GetTitle, True, lngErr
- SW.CloseDoc objDoc.GetTitle

Removing a feature pattern typically involves selecting the derived pattern from the parent model document and deleting the selected derived pattern. The following API functions may be used:

- oParentDoc.ClearSelection
- oParentDoc.SelectByID strPatternName, "BODYFEATURE", 0, 0, 0
- oParentDoc.DeleteSelection False
- Assembly.EditRebuild3

Setting a custom property typically involves verifying the value and setting the value if necessary. The following API functions may be used:

- objDoc.GetCustomInfoValue("", strName) <> varArgument
- objDoc.CustomInfo2("", strName) = varArgument
- Assembly.EditRebuild 3

Setting a component color typically involves retrieving the MaterialPropertyValues for the component from the component or the model document, changing the first three elements in the array to reflect the new color, and setting the new

MaterialPropertyValues on the component. If the color is being removed, the RemoveMaterialProperty is called. The following API functions may be used:

- arr = objComponent.MaterialPropertyValues()
- arr = objComponent.GetModelDoc.MaterialPropertyValues()
- objComponent.MaterialPropertyValues = arr
- Assembly.EditRebuild 3
- objComponent.RemoveMaterialProperty

Closing SOLIDWORKS(TM) typically involves closing the top level assembly (which is typically the only one open at this point) and calling the Exit App API function. The following API functions may be used:

- SW.QuitDoc strTitle
- SW.ExitApp

Various aspects of an exemplary embodiment of the present invention are described hereinafter with reference to modeling a fan. In accordance with an embodiment of the present invention, a fan includes various sub-parts, including a fan assembly, a housing assembly, a motor assembly, and a mounting assembly. Certain components of the fan might have fixed characteristics. For example, a company might purchase three different motors that can be used in a fan, and these motors have fixed dimensions that cannot be changed in the model. CAD models of the motors may be created in the CAD system and imported into the knowledge management application for storage by the knowledge storage application. Other components of the fan might have characteristics that can be determined dynamically during modeling. For example, the dimensions of a fan hub might depend on the motor selected for the model. Models of these components might be created in the CAD system and imported into the knowledge management application, or rules for defining these components might be established. For purposes of the following example, the CAD system is presumed to be the SOLIDWORKS(TM) three-dimensional CAD system.

FIG. 5 shows an exemplary user interface screenshot 500 for importing information from the CAD system relating to a mounting assembly, such as might be generated by the knowledge management application in accordance with an embodiment of the present invention. The screenshot 500 displays, among other things, a hierarchical part family tree 510 showing that a BuildingServicesFan part family includes sub-parts entitled FanAssembly, HousingAssembly, MotorAssembly, and MountingAssembly, with the sub-parts having their own part families entitled Rotor, Housing, Motor, and Mounting, respectively. The screenshot 500 shows information relating to the MountingAssembly sub-part (as indicated by the MountingAssembly sub-part being highlighted in the part family tree 510), including a list of all specifications 520 relating to the MountingAssembly sub-part and a window 530 for entering information about the MountingAssembly sub-part. The screenshot 500 also includes a toolbar 540 from which various functions of the knowledge management application (such as creating a new part family, sub-part, property, connection, or CAD specification) can be accessed using either pull-down menus or icons. The window 530 includes a portion 531 showing that there is a single valid part family entitled Mounting associated with the MountingAssembly sub-part. It should be noted that there could be multiple part families associated with the MountingAssembly sub-part, and all valid part families would be displayed in the portion 531. The window 530 also includes a portion 532 for defining a rule to determine the optimal part family for a particular model (in this case, the optimal part family is the Mounting part family by default).

In order to associate a mounting component defined in the CAD system with the Mounting part family, the user might highlight "Mounting" in the part family tree 510 and then select a function from the toolbar 540 to create a new geometric specification (this function can be accessed from either the File menu or an icon on the toolbar). FIG. 6 shows an exemplary user interface screenshot 600 for defining a new geometric specification relating to the Mounting part family, such as might be generated by the knowledge management application in accordance with an embodiment of the present invention. The screenshot 600 shows information relating to the Mounting part family of the MountingAssembly sub-part (as indicated by the Mounting part family being highlighted in the part family tree 610), including a list of all specifications 620 relating

to the Mounting part family and a window 630 for entering the new geometric specification for the Mounting part family. The window 630 shows the geometry type 631 (in this case, "SolidWorks") and a list of valid CAD part files 632 associated with the Mounting part family (in this case, none have yet been specified).

In order to associate a CAD part file with the Mounting part family, the user might select the "add part file" control 633 and enter the name of a CAD part file, in which case the knowledge management application imports from the CAD system information relating to the specified CAD part file. The knowledge management application preferably displays a list of parameters defined for the part in the CAD part file. FIG. 7 shows an exemplary user interface screenshot 700 displaying information upon entry of the name of a new CAD part file, such as might be generated by the knowledge management application in accordance with an embodiment of the present invention. The screenshot 700 shows the part family tree 710, a list of all specifications 720 relating to the Mounting part family, a window 730 showing the new valid part file (Mounting-Aero.SLDPR), and a window 740 displaying a list of parameters defined for the part in the CAD part file.

Once the CAD part file has been associated with the Mounting part family, the user typically defines various geometry features and associates the geometry features with specific CAD parts. In order to define a geometry feature and associate the geometry feature with one or more specific CAD parts, the user might select a function from the toolbar 750 to create a new geometry feature (this function can be accessed from either the File menu or an icon on the toolbar). FIG. 8 shows an exemplary user interface screenshot 800 for defining a geometry feature, such as might be generated by the knowledge management application in accordance with an embodiment of the present invention. The screenshot 800 includes a window 840 for defining a geometry feature and associating the geometry feature with one or more specific CAD parts. In this case, a geometry feature having a display name 842 HubDiameter and a system name HubDiameter 843 is associated with a corresponding CAD part entitled HubDiameter by specifying a formula in portion 844.

FIG. 9 shows an exemplary user interface screenshot 900 showing geometry features (properties) defined for the CAD part file, such as might be generated by the

knowledge management application in accordance with an embodiment of the present invention. The screenshot 900 shows the part family tree 910, a list of all specifications 920 relating to the Mounting part family, and a window 930 including a properties portion 931 showing the geometry features associated with specific CAD parts (in this case, a HubDiameter geometry feature and a MountingDiameter geometry feature).

After the geometry features have been defined and associated with corresponding CAD parts, the user typically defines any mating rules associated with the CAD parts file. In order to define a mating, the user may select a function from the toolbar 950 (this function can be accessed from either the File menu or an icon on the toolbar). FIG. 10 shows an exemplary user interface screenshot 1000 for defining a mating, such as might be generated by the knowledge management application in accordance with an embodiment of the present invention. The screenshot 1000 shows the part family tree 1010, a list of all specifications 1020 relating to the Mounting part family, a part file window 1030, and a window 1040 for entering mating information.

FIG. 11 shows an exemplary user interface screenshot 1100 showing mating definitions, such as might be generated by the knowledge management application in accordance with an embodiment of the present invention. The screenshot 1100 shows the part family tree 1110, a list of all specifications 1120 relating to the Mounting part family, and a window 1130 displaying various mating (orientation) definitions 1131 for the CAD part.

As discussed above, the user can establish rules for attributes that may be difficult or impossible to model in the CAD system, including certain geometric attributes (such as "negative" attributes), certain geometry-based attributes that are not modeled as physical features of a structure or assembly, and non-geometric attributes (such as pricing and processes). For example, continuing with the fan modeling scenario above, it might be useful to establish a rule for computing a non-modeled geometry-based attribute, such as area covered by the sweep of the fan blade component when rotating. While this fan area might be useful, for example, for computing the volume of air moved by a fan blade component having a specified number of blades rotating at a specified rate, the fan area is typically not modeled as a physical feature of the fan. FIG. 12 shows an exemplary user interface screenshot 1200 showing a rule for determining a fan area for the fan, such as

might be generated by the knowledge management application in accordance with an embodiment of the present invention. The screenshot 1200 shows the part family tree 1210 (with the part family BuildingServicesFan highlighted), a list of all specifications 1220 relating to the BuildingServicesFan part family (with the FanArea specification highlighted), and a window 1230 displaying the rule 1240 for determining the fan area based on the diameter of the fan blade component. The user can establish other rules that utilize this FanArea value.

After component information has been imported from the CAD system and modeling rules have been established, the knowledge management application controls the CAD system through its API to produce a geometric model according to predetermined specifications. The specifications can be incorporated into the rules and/or provided by a user at run time. The CAD model may be produced for display to the user via the graphical user interface, or may be produced solely for the knowledge management application to obtain model-related information from the CAD system. As an example of the former, the knowledge management application can control the CAD system to generate a model and then produce a graphical display through the graphical user interface including a graphical representation of the model as generated by the CAD system (e.g., by displaying a display window generated by the computer-aided design system), with or without related information from the knowledge management system. As an example of the latter, the knowledge management application can control the CAD system to generate a model and then control the CAD system to compute the surface area of a component for a cost estimate to be produced by the knowledge management application.

FIG. 13 shows an exemplary user interface screenshot 1300 including an embedded graphical representation of a model generated by the CAD system, such as might be generated by the knowledge management application in accordance with an embodiment of the present invention. The screenshot 1300 shows the part family tree 1310 (with the part family BuildingServicesFan highlighted), a list of all specifications 1320 relating to the BuildingServicesFan part family, and a window 1330 including a portion 1340 including a graphical representation of the model generated by the CAD system.

The user can specify how information is to be displayed by the knowledge management application. For example, the user can specify that the CAD system window be displayed under some conditions but not others, and can also specify what knowledge management system information to display along with the CAD system window.

FIG. 14 shows a first exemplary user interface screenshot 1400 including a sub-window generated by the CAD system, such as might be generated by the knowledge management application in accordance with an embodiment of the present invention. The screenshot 1400 displays a function list 1410, information from the knowledge management system 1430, and a sub-window 1440 generated by the CAD system including a graphical representation of the model generated by the CAD system and controls for manipulating the model.

FIG. 15 shows a second exemplary user interface screenshot 1500 including a sub-window generated by the CAD system, such as might be generated by the knowledge management application in accordance with an embodiment of the present invention. The screenshot 1500 displays a function list 1510, a part family tree 1520, information from the knowledge management system 1530, a sub-window 1540 generated by the CAD system including a graphical representation of the model generated by the CAD system and controls for manipulating the model, and properties information 1550.

FIG. 16 shows a third exemplary user interface screenshot 1600 including a full view window generated by the CAD system, such as might be generated by the knowledge management application in accordance with an embodiment of the present invention. The screenshot 1600 displays a function list 1610 and a full view window 1630 generated by the CAD system.

In certain embodiments of the invention, the user can make rule changes on the fly, and the knowledge management application will control the CAD system to update the model accordingly and will display the updated graphical representation of the model to the user substantially in real time. In this way, the user essentially gets immediate feedback regarding the rule change.

In certain embodiments of the invention, changes to a model can be made in the CAD system, and the knowledge management application will identify those changes

through interactions with the CAD system and will modify and apply rules accordingly. For example, if the user makes a change in the CAD system that overrides a particular rule, the knowledge management application might cause appropriate tracking information to be stored by the knowledge storage application, create one or more revised rules that reflect the change, and apply other rules to update other components of the model according to the rules. The manner in which the knowledge management application can identify CAD system changes depends to a large degree on the CAD system API. For example, the CAD system might communicate the changes to the knowledge management application, or the knowledge management application might monitor or poll the CAD system for changes.

In certain embodiments of the invention, the knowledge management application can cause a particular model part displayed in the CAD system window to be displayed or highlighted when the user is working on rules relating to that part. For example, with reference again to FIG. 13, if the user selects the Motor part family in the part family tree 1310, the knowledge management application might cause the motor to be highlighted in the window 1340, for example, by changing the color of the motor.

In certain embodiments of the invention, the knowledge management application can cause information relating to a particular model part to be displayed when the user highlights that part in the CAD system window. For example, with reference again to FIG. 13, if the user highlights the fan blade component in the CAD window 1340, the knowledge management application might cause information relating to the fan blade component to be displayed in the window 1330, with appropriate highlighting in the part family tree 1310 and the list of specifications 1320.

As discussed above, the knowledge management application can interoperate with two-dimensional CAD systems as well as three-dimensional CAD systems. In an exemplary embodiment of the present invention, the knowledge management application supports both the two-dimensional CAD system VISIO(TM) and the three-dimensional CAD system SOLIDWORKS(TM).

FIG. 17 shows an exemplary user interface screenshot 1700 including a window generated by a two-dimensional CAD system, such as might be generated by the knowledge management application in accordance with an embodiment of the present

invention. The screenshot 1700 displays a function list 1710, a part family tree 1720, a CAD system window 1740, and properties information 1750.

The knowledge management application can also interoperate with various analysis applications. Typically, the knowledge management application exports information to the analysis application for analysis. The knowledge management application can display analysis information to the user.

FIG. 18 shows an exemplary user interface screenshot 1800 including an analysis window, such as might be generated by the knowledge management application in accordance with an embodiment of the present invention. The screenshot 1800 displays a function list 1810, a part family tree 1820, a CAD system window 1840, properties information 1850, and an analysis window 1860 generated by the analysis application.

Thus, in certain embodiments of the present invention, structures are defined in a CAD system and are associated with a structure class. Rules are defined for the structure class. When one of the structures is selected by the knowledge management system for a computer-aided design model, the rules are applied to the selected structure. These class-based rules make it easier for the user to define rules, since a single rule defined by the user gets applied to an entire class of structures, and therefore the user does not have to define the rule individually for each structure of the structure class.

FIG. 21 is a logic flow diagram showing exemplary logic 2100 for class-based rules in accordance with an embodiment of the present invention. Starting in block 2102, structures (such as parts for an assembly) are defined in a CAD system, in block 2104. The structures are associated with a structure class (such as a part family) in a knowledge management system, in block 2106. At least one rule is defined that applies to the structure class, in block 2108. When one of the structures is selected for a computer-aided design model by the knowledge management system, in block 2110, the knowledge management system applies the rule(s) to the selected structure, in block 2112. The logic 2100 ends in block 2199.

As discussed above, the knowledge management application obtains a set of rules from a central database. The set of rules may include rules relating to geometric and non-geometric attributes. The non-geometric attributes may be dependent on geometric attributes. The knowledge management application generates instructions for modeling a

geometric structure based on the set of rules. The knowledge management application communicates the instructions to a computer-aided design system, typically through an API of the computer-aided design system. The knowledge management application may produce a graphical display on a graphical user interface including information from the knowledge management system as well as information from the computer-aided design system (such as a graphical representation of a geometric model).

FIG. 22 is a logic flow diagram showing exemplary logic 2200 for the knowledge management application in accordance with an embodiment of the present invention. Starting in block 2202, the logic obtains a set of rules from a central database, in block 2204. The logic generates instructions for modeling a geometric structure based on the set of rules, in block 2206. The logic communicates the instructions to a computer-aided design system, in block 2208, typically through an application program interface of the computer-aided design system. The logic may produce a graphical display on a graphical user interface including a first portion including information from the knowledge management application and a second portion including information from the computer-aided design system, in block 2210. The logic ends in block 2299.

It should be noted that the logic flow diagrams are used herein to demonstrate various aspects of the invention, and should not be construed to limit the present invention to any particular logic flow or logic implementation. The described logic may be partitioned into different logic blocks (e.g., programs, modules, functions, or subroutines) without changing the overall results or otherwise departing from the true scope of the invention. Often times, logic elements may be added, modified, omitted, performed in a different order, or implemented using different logic constructs (e.g., logic gates, looping primitives, conditional logic, and other logic constructs) without changing the overall results or otherwise departing from the true scope of the invention.

The present invention may be embodied in many different forms, including, but in no way limited to, computer program logic for use with a processor (*e.g.*, a microprocessor, microcontroller, digital signal processor, or general purpose computer), programmable logic for use with a programmable logic device (*e.g.*, a Field Programmable Gate Array (FPGA) or other PLD), discrete components, integrated circuitry (*e.g.*, an Application Specific Integrated Circuit (ASIC)), or any other means

including any combination thereof. In a typical embodiment of the present invention, predominantly all of the knowledge management system applications are implemented as a set of computer program instructions that are executed by a computer under the control of an operating system.

Computer program logic implementing all or part of the functionality previously described herein may be embodied in various forms, including, but in no way limited to, a source code form, a computer executable form, and various intermediate forms (*e.g.*, forms generated by an assembler, compiler, linker, or locator). Source code may include a series of computer program instructions implemented in any of various programming languages (e.g., an object code, an assembly language, or a high-level language such as Fortran, C, C++, JAVA, or HTML) for use with various operating systems or operating environments. The source code may define and use various data structures and communication messages. The source code may be in a computer executable form (*e.g.*, via an interpreter), or the source code may be converted (*e.g.*, via a translator, assembler, or compiler) into a computer executable form.

The computer program may be fixed in any form (*e.g.*, source code form, computer executable form, or an intermediate form) either permanently or transitorily in a tangible storage medium, such as a semiconductor memory device (*e.g.*, a RAM, ROM, PROM, EEPROM, or Flash-Programmable RAM), a magnetic memory device (*e.g.*, a diskette or fixed disk), an optical memory device (*e.g.*, a CD-ROM), a PC card (*e.g.*, PCMCIA card), or other memory device. The computer program may be fixed in any form in a signal that is transmittable to a computer using any of various communication technologies, including, but in no way limited to, analog technologies, digital technologies, optical technologies, wireless technologies (*e.g.*, Bluetooth), networking technologies, and internetworking technologies. The computer program may be distributed in any form as a removable storage medium with accompanying printed or electronic documentation (*e.g.*, shrink wrapped software), preloaded with a computer system (*e.g.*, on system ROM or fixed disk), or distributed from a server or electronic bulletin board over the communication system (*e.g.*, the Internet or World Wide Web).

Hardware logic (including programmable logic for use with a programmable logic device) implementing all or part of the functionality previously described herein may be

designed using traditional manual methods, or may be designed, captured, simulated, or documented electronically using various tools, such as Computer Aided Design (CAD), a hardware description language (*e.g.*, VHDL or AHDL), or a PLD programming language (e.g., PALASM, ABEL, or CUPL).

Programmable logic may be fixed either permanently or transitorily in a tangible storage medium, such as a semiconductor memory device (*e.g.*, a RAM, ROM, PROM, EEPROM, or Flash-Programmable RAM), a magnetic memory device (*e.g.*, a diskette or fixed disk), an optical memory device (*e.g.*, a CD-ROM), or other memory device. The programmable logic may be fixed in a signal that is transmittable to a computer using any of various communication technologies, including, but in no way limited to, analog technologies, digital technologies, optical technologies, wireless technologies (*e.g.*, Bluetooth), networking technologies, and internetworking technologies. The programmable logic may be distributed as a removable storage medium with accompanying printed or electronic documentation (*e.g.*, shrink wrapped software), preloaded with a computer system (*e.g.*, on system ROM or fixed disk), or distributed from a server or electronic bulletin board over the communication system (*e.g.*, the Internet or World Wide Web).

The present invention may be embodied in other specific forms without departing from the true scope of the invention. The described embodiments are to be considered in all respects only as illustrative and not restrictive.